Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anej Lekše

# Avtomatizacija delavniškega dnevnika

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Andrej Brodnik

Ljubljana, 2020

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preveri ali so glasovni asistenti v trenutnem stanju primerni za pomoč pri pisanju delavniških dnevnikov. Seznani se z obstoječimi programskimi rešitvami in jih analiziraj. Po analizi trga se loti izdelave svojega sistema za pisanje delavniških dnevnikov, ki vključuje glasovnega asistenta.

*Na tem mestu zapišite, komu se zahvaljujete za izdelavo diplomske naloge. Pazite, da ne boste koga pozabili. Utegnil vam bo zameriti. Temu se da izogniti tako, da celotno zahvalo izpustite.*

# Kazalo

# Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|---|---|---|
| **API** | Application Programming Interface | vmesnik za programiranje |
| **AWS** | Amazon Web Services | Amazonove spletne storitve |
| **LIMS** | Laboratory Information Management System | laboratorijski sistem za upravljanje informacij |
| **SNS** | Simple Notification Service | preprosta storitev za opozorila |
| **SQS** | Simple Queue Service | preprosta vrstna storitev |
| **UI** | User Interface | uporabniški vmesnik |
| **VUI** | Voice User Interface | glasovni uporabniški vmesnik |
| **XAML** | Extensible Application Markup Langugage | razširljiv aplikacijski označitveni? jezik |
| **XAML** | Extensible Application Markup Langugage | razširljiv aplikacijski označitveni? jezik |

# Povzetek

**Naslov:** Avtomatizacija delavniškega dnevnika

**Avtor:** Anej Lekše

Diplomsko delo obravnava področje izboljšanja procesa pisanja delavniškega dnevnika ali laboratorijskega poročila. Največ časa anketirani študentje porabijo za prepisovanje v digitalno obliko, zapisovanje zapiskov na papir in urejanje teh zapiskov. Cilj je preizkusiti računalniški sistem z glasovnim asistentom, s pomočjo katerega lahko narekujemo zapiske med delom. Te zapiske pa lahko naknandno urejamo s pomočjo mobilne ali namizne aplikacije. Uporabili bomo Amazon Alexo zaradi enostavne izdelave lastnih programov (t.i. Skill-ov).

**Ključne besede:** mobilni razvoj, glasovni asistenti, razpoznava glasu, informacijski sistemi.

# Abstract

**Title:** Workshop report automatisation

**Author:** Anej Lekše

This thesis deals with the process of optimising the process of writing a lab report. Students, that took part in the survey, spend the most time to type the report into a digital format, write notes on paper and ordering their notes. The goal of thesis is testing a system with a voice assistant that could be used to take notes during work itself. These notes can be edited and ordered later via a mobile or desktop application. We will use Amazon Alexa as it offers simple programming with Alexa Skills.

**Keywords:** mobile development, voice assistants, voice recognition, information systems.

# Poglavje 1

# Uvod

## 1.1 Opis domene raziskave

Diplomsko delo obravnava področje pisanja delavniških in laboratorijskih poročil. Delavniški dnevnik je dokument, ki opisuje potek izdelave izdelka po korakih. Zapis koraka dela vsebuje opis dela, uporabljena orodja in metode ter trajanje. Delavniški dnevnik lahko opisuje tudi korake kontrolnega postopka za željen izdelek.

V diplomski nalogi smo želeli raziskati, ali se da za učinkovitejše pisanje delavniškega dnevnika uporabiti računalniško rešitev. Sistem, s katerim bi to testirali, bi sestavljali glasovni asistent, ki bi služil za narekovanje opomb, mobilne aplikacije, preko katere bi lahko urejali zapiske in strežnika.

## 1.2 Struktura diplomske naloge

Diplomsko delo pričenjamo s predstavitvijo področja raziskave in kratko opišemo problem in možno rešitev. V sklopu te diplomski naloge bomo raziskali, ali je glasovni asistent in mobilna aplikacija primerno orodje za učinkovitejše pisanje delavniških dnevnikov. Začnemo z raziskavo trga za že obstoječe rešitve za ta problem. Nato opišemo, kaj trenutne tehnologije tega področja ponujajo in njihove močne ter šibke točke. Po analizi se lotimo opisa

tehnologij, ki smo jih pri pisanju diplome uporabili. V naslednjem poglavju se lotimo načrtovanja sistema za pomoč pri pisanju laboratorijskih poročil. Natančno definiramo funkcionalnosti sistema, utemeljimo odločitev za izbiro Amazon Alexe in izdelamo Alexa Skill, API in mobilno aplikacijo. Funkcionalnosti sistema OpenReport testiramo in analiziramo. V predzadnjem poglavju opišemo možnosti nadaljnjega razvoja projekta.

# Poglavje 2

# Pregled problema in rešitve

## 2.1 Problem

Pisanje delavniškega dnevnika je lahko zamudno delo. Velik problem predstavlja razdrobljenost podatkov - opombe o delu se nahajajo v beležkah ali na listih papirja, fotografije v pomnilniku mobilnih naprav, itd. Poleg tega je treba podatke po koncu dela smiselno urediti.

Izhajajoč iz navedenega opredeljujem problem diplomskega dela: kako lahko proces izdelave delavniškega dnevnika naredimo učinkovitejši in prijaznejši uporabniku?

## 2.2 Analiza trga

### 2.2.1 Kaj ponuja trg?

Za pisanje laboratorijskih poročil in delavniških dnevnikov ne obstaja veliko rešitev, namenjenih izključno tej domeni. Velikokrat so vključene v različne LIMS sisteme. Ti sistemi so obsežni, saj poleg možnosti pisanja poročil, vključujejo še podsisteme za upravljanje z eksperimenti, zaposlenimi, naročevanje opreme itd.

Sistemi, ki se bolj osredotočajo na pisanje poročil pa praviloma dajejo

večji poudarek prikazovanju podatkov in risanju grafov.

Program, ki je prišel najbližje rešitvi problema te naloge je bil program Squid. Zapiske je enostavno pisati na tablični računalnik in jih izvoziti kot PDF. Slike se nahajajo na isti platformi kot poročilo.

Kljub temu tudi pri Squid-u ostaja problem prepisovanja na roko napisanega besedila. Kar je napisano na roko je namreč treba pretipkati, če želimo besedilo urejati. Poleg tega besedila ne moremo narekovati z glasom.

Najbolj zanimiva rešitev tega problema se nam je zdela dodelava medicinskega laboratorija s tehnologijo IoT Namen raziskave je bil preizkus praktične uporabnosti virtualnih asistentov za naloge, kot so branje laboratorijskih postopkov po korakih, in glasovno uprabljanje laboratorijskih instrumentov. Pozitivni rezultati bi lahko bili ključnega pomena za slabovidne člane laboratorijev. Kot glasovni asistent je bila uporabljena Amazon Alexa. Raziskovalna ekipa je v laboratorij lahko vgradila IoT naprave brez previsokih stroškov. Prepoznavanje govora in ukazov je bilo konsistentno in hitro, ne glede na spol uporabnika. Motnje pri razpoznavanju je povzročal večinoma ozadni hrup. Povprečna natančnost prepoznave ukazov je bila 95%. Raziskovalci so zabeležili tudi problem moteče kakofonije v laboratoriju, v katerem je več raziskovalcev, ki uporabljajo glasoven nadzor naprav.

### 2.2.2 Kako se pisanja poročil lotevajo študentje?

Ker so delavniški dnevniki zelo podobni laboratorijskim poročilom, sva z mentorjem sklenila, da o postopku pisanja laboratorijskih poročil povprašam študente biologije in kemije na sosednjih fakultetah.

95% izprašanih študentov za pisanje poročil uporablja prenosne računalnike in pisarniške programe (od teh 90% Microsoft Word, 10% odprtokodne programe OpenOffice ali LibreOffice Writer). Manjši delež študentov (5%) pa uporablja tablični računalnik in program Squid.

Največja problema jim predstavljata prenašanje datotek med napravami in dolgotrajno pisanje besedilnih del poročila.

## 2.3 Rešitev

Po premisleku in raziskavi trga sem prišel do ugotovitve, da učinkovit program za pisanje poročila:

- povzroča čim krajše prekinitve dela,
- ima čim preprostejši uporabniški vmesnik,
- vsebuje možnost uporabe glasu za narekovanje opomb,
- hrani uporabljene podatke (fotografije, zapiske) na enem mestu.

Predvideno rešitev za zadani problem sem poimenoval OpenReport. Sistem OpenReport bodo sestavljali:

- strežniški program,
- mobilna aplikacija,
- glasovni asistent.

Vsak del sistema bo opravljal svojo nalogo. Strežniški program bo tekel na osebnem računalniku, in bo ponujal API, ki ga bosta uporabljala glasovni asistent in mobilna aplikacija. Strežniški program bo tudi skrbel za hranjenje podatkov na računalniku, na katerem bo tekel. Skrbel bo tudi za izvoz poročila v tekstovni obliki ali v obliki dokumenta.

Mobilna aplikacija za operacijski sistem Android bo osnovni način prikazovanja elementov delavniškega dnevnika. Preko te aplikacije bo uporabnik lahko:

- ustvaril nova poročila,
- odprl obstoječa poročila,
- zajemal slike in jih vstavljal v odprto poročilo,
- napisal tekstovne opombe in jih vstavljal v poročilo,

- urejal vrstni red elementov poročila.

Glasovni asistent bo služil za narekovanje opomb. Za glasovnega asistenta bom uporabil Amazon Alexo.

# Poglavje 3

# Tehnologije in programska oprema

## 3.1 Visual Studio 2019

Visual Studio 2019 je integrirano razvojno okolje (IDE), ki ga je razvil Microsoft. Tesno je integriran z Microsoftovimi tehnologijami, kot so npr. .NET, Xamarin, itd. Pri našem delu sem uporabljal Community izdajo Visual Studia 2019.

## 3.2 .NET

.NET je razvojna platforma, razvita s strani Microsofta. Obsega programske jezike, prevajalnike, orodja in knjižnice, ki omogočajo širok spekter primerov uporabnosti, hkrati pa se ohranja enovitost ozadne kode.

Najpomembnejše tehnologije, ki ga sestavljajo so:

- .NET Framework - platforma za razvoj spletnih strani in namiznih Windows aplikacij,
- .NET Core - odprtokodna platforma za razvoj cross-platform aplikacij in spletnih strani,

- Xamarin, ki je ogrodje za razvoj mobilnih aplikacij za najpogostejše mobilne operacijske sisteme (Android, iOS).

Poleg teh tehnologij je .NET vključen tudi v razna druga orodja. Uporabljajo se za za razvoj iger, virtualne resničnosti, računalniškega vida, strojnega učenja, itd.

## 3.3 Xamarin Forms

Ogrodje Xamarin je odprtokodno orodje za razvoj mobilnih aplikacij, ki ga je razvil Microsoft. Je nadgradnja ogrodja .NET.

Z njim je mogoče pri preprostih aplikacijah deliti večino kode med različnimi operacijskimi sistemi. Za ozadno kodo se uporablja .NET (C#), za front-end pa se uporablja XAML (Extensible Application Markup Language).

Pri razvoju sem poudarek dal delu Xamarin ogrodja, ki se imenuje Xamarin.Forms Shell (v nadaljevanju *Shell*).

Osnovna naloga Shell-a je, da med različnimi mobilnimi operacijskimi sistemi poenoti osnovne elemente uporabniških vmesnikov. Poenotijo se tudi nekatere druge funkcije, ki jih večina mobilnih aplikacij potrebuje. Med te funkcije spadajo navigacija po straneh, URI navigacijska shema, iskalnik in nekaj najpogostejših tipov postavitve aplikacije (Flyout, Tabs, itd.).

Kodo za sam izgled aplikacije (v nadaljevanju *front-end*) se piše v jeziku XAML. XAML je po strukturi zelo podoben XML ali HTML. Grafične ali logične elemente se definira v ščipastih oklepajih "<>". Ti tvorijo elemente, ki se jih da dopolnjevati, gnezditi, nizati, itd.

V Xamarinovi različici XAML-a so grafični elementi različnih mobilnih platform, ki imajo podobne funkcije združeni pod enotnimi imeni. V ozadju lahko tem elementom v C# razredih prirejamo vrednosti.

## 3.4 Amazon Alexa

Amazon Alexa je virtualni asistent, razvit s strani podjetja Amazon. Prvič je bila uporabljena v pametnih zvočnikih Amazon Echo. Z Alexo lahko predvajamo glasbo, radijske oddaje, nastavljamo alarme, itd. Alexa lahko nadzira tudi pametne gospodinjske naprave.

### 3.4.1 Alexa Skill

Alexine osnovne funkcionalnosti lahko nadgradimo s funkcijami, ki se jim reče Skill-i. Skill lahko naredi vsakdo z Amazon Developer računom.

Skill sestavljajo:

- Invocation - fraza, ki Skill zažene,
- Intent - fraze, ki jih Skill razpozna kot funkcije,
- Endpoint - omrežni vir, kjer se nahaja ozadna koda Skill-a.

Skill je vezan na Amazonov uporabniški račun. Ko Alexa zasliši Invocation ali katerega od Intentov, glasovni posnetek pošlje na Amazonov strežnik. Ta s pomočjo strojnega učenja razpozna ukaze in pošlje ukaze na Endpoint. To je lahko druga Amazonova storitev (npr. AWS Lambda), Microsoftov Azure strežnik ali naš lasten strežnik. Ko Endpoint obdela zahtevo, se rezultat pošlje nazaj na Amazonov strežnik v obliki znakovnega niza. Ta podatek se nato pošlje nazaj na uporabnikovo Alexo, ki prejeti znakovni niz izgovori.

## 3.5 Amazon Web Services

AWS je skupek oblačnih storitev, ki ga ponuja podjetje Amazon. AWS sestavlja več kot 200 storitev, od gostovanja spletnih strani, do sistemov za pošiljanje in prejemanje obvestil, sistemov za komunikacijo med storitvami, itd. Ponuja dodelano integracijo s popularnimi programskimi jeziki in ogrodji, kot so Java, .NET, Python, Node.js, itd. Storitev je plačljiva

po principu "pay-as-you-go"ali "plačaj kolikor porabiš". Pri tej diplomski nalogi se bom osredotočil na uporabo dveh sistemov.

### 3.5.1 AWS SQS

AWS SQS je sistem za pošiljanje tekstovnih sporočil med odjemalci preko Amazonovih strežnikov. Za hranjenje sporočil je treba registrirati Queue ali vrsto. Ta je lahko navadna vrsta, kjer prejeta sporočila niso nujno urejena po času ustvarjanja, lahko pa je tipa FIFO. Pri FIFO vrsti sporočila prejmemo v točno takšnem vrstnem redu, kot smo jih poslali. Da pošljemo sporočilo pošljemo Šend Message Request". Da sporočilo prejmemo pošljemo zahtevo "Recieve Message Request". Ker lahko iz vrste poberemo največ 5 sporočil naenkrat, moramo prejeta sporočila iz nje tudi izbrisati. To dosežemo z "Delete Message Request".

### 3.5.2 AWS Lambda

AWS Lambda je dogodkovno-vodena računalniška platforma, razvita s strani Amazona. Poganja programske funkcije na podlagi prejetih dogodkov in avtomatsko razpolaga z računalniškimi viri, na podlagi zahtevnosti operacij. Storitev je bila ustvarjena predvsem za učinkovito posodabljanje DynamoDB podatkovnih baz in nalaganje slik in objektov na storitve, kot so Amazon S3.

# Poglavje 4

# Načrtovanje in razvoj sistema OpenReport

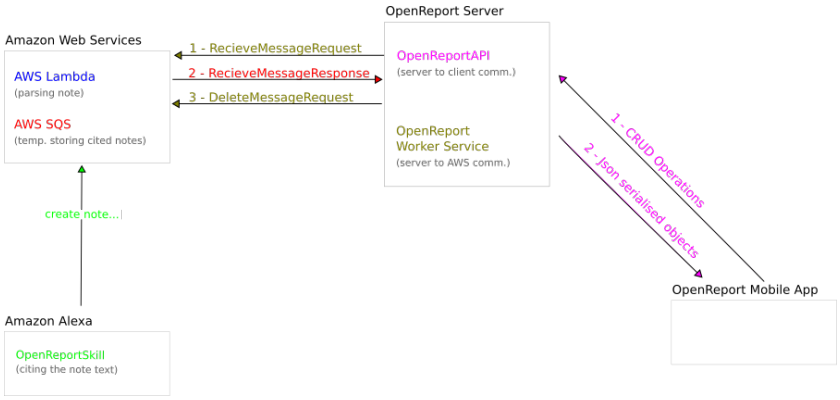
## 4.1 Načrt

Sistem OpenReport sestavljajo strežnik, glasovni asistent, mobilna aplikacija in skupek AWS storitev (Slika **??**).

Glasovni asistent Amazon Alexa ima nalogo sprejemati glasovna navodila uporabnika. Ko od uporabnika prejme zapisek, OpenReportSkill ta zapisek pošlje v AWS SQS vrsto. Alexa uporabniku javi, da je sporočilo hranjeno.

Naloga strežnika je, da hrani podatke o uporabnikih, projektih in zapiskih. Strežnik sestavljata dve storitvi. Prva je API, ki ponuja vmesnik za upravljanje s podatki v podatkovni bazi. Druga storitev je delavec, ki v določenih časovnih intervalih kontaktira storitev AWS SQS vrsto, v katero Alexa zapisuje opombe. V kolikor je v tej vrsti sporočilo ga prejme, vpiše v podatkovno bazo in ga izbriše iz SQS vrste.

Mobilna aplikacija kontaktira API.




Slika 4.1: Visokonivojski načrt sistema

## 4.2 Definicija funkcionalnosti

Na začetku smo definirali primere uporabe sistema OpenReport. Uporabnik mora imeti možnosti ustvariti nov projekt ali odpreti obstoječega. Projekt lahko uporabnik tudi izbriše ali arhivira.

Uporabnik lahko projektu doda zapiske, ki so lahko v besedilni obliki ali slike. Te lahko doda preko mobilne aplikacije ali jih narekuje s pomočjo glasovnega asistenta.

Uporabnik lahko ureja zapiske, spreminja njihov vrstni red ali jih odstranjuje.

## 4.3 Odločitev za izbor Amazon Alexe

Za Amazon Alexo sem se odločil, saj ponuja enostavno možnost programiranja s Skilli. To poleg tega za testiranje ne rabimo fizične enote, saj si lahko na vsako Android napravo namestimo Alexa aplikacijo. Aplikacija ponuja iste funkcionalnosti kot pametni zvočniki Amazon Echo ali Amazon Dot. Poleg tega jo je zelo enostavno spojiti z drugimi Amazonovimi spletnimi storitvami, kot so AWS SQS ali AWS SNS.

## 4.4 Načrt delovanja strežnika

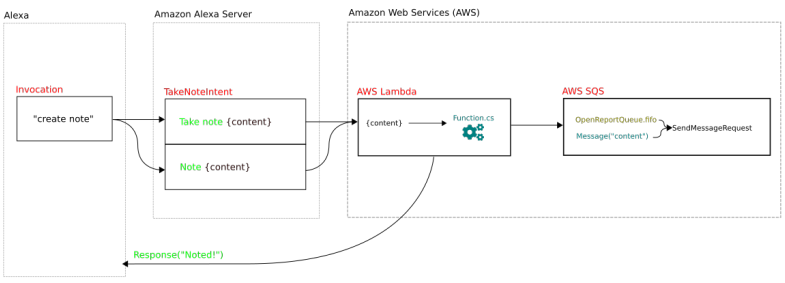
API endpointi

## 4.5 Načrt in izdelava Alexa Skilla

OpenReportSkill se bo začel izvajati z invokacijo črate note". Uporabnik mora nato začeti naslednji korak s "take note"in narekovati besedilo opombe. Alexa pošlje posnetek govora na Amazonov Alexa Server, kjer se zahteva obdela in primerja z nastavljenimi Intenti. Frazi "note"ali "take note", na začetku narekovanega besedila, se ujemata s frazami za klic Intenta TakeNoteIntent.

Amazonov Alexa Server nato pošlje TakeNoteIntent in vsebino (na sliki **??** {content}) na nastavljen Endpoint. Ta je v našem primeru gostovan na storitvi AWS Lambda.

Endpoint gosti ozadno kodo za obdelavo podatkov, ki jih Alexa Server pošlje. V našem Endpointu se TakeNoteIntent obdela tako, da se {content} pošlje kot novo sporočilo v AWS SQS. Ko je sporočilo poslano, Endpoint Alexa Serverju pošlje nazaj zahtevo Response. Ta hrani niz, ki ga bo Alexa izgovorila uporabniku. V našem primeru je to stavek "Noted!".




Slika 4.2: Načrt Alexa Skilla

## 4.6 Izdelava API

## 4.7 Izdelava delavca

Delavec (ang. Worker Service) je storitev, ki se praviloma konstatno izvaja v ozadju. V našem primeru se delavec uporablja za dobivanje sporočil iz AWS SQS sporočilne vrste in vstavljanje prejetih sporočil v podatkovno bazo.

Ustvaril sem ga tako, da sem v trenuten Solution dodal nov projekt tipa WorkerService. Komponente takega projekta so predvsem razred Startup, ki se kliče prvi po zagonu programa in razred Program, v katerega pišemo poslovno logiko programa.

Pomemben del programiranja delavca je bil konfiguracija AWS storitev. Zaradi varnosti, v projektne datoteke nisem neposredno vpisal vseh dostopnih podatkov za Amazonove spletne vire. Najvarnejši način za dostopanje do teh podatkov je uporaba privzete AWS CLI lokacije za hranjenje dostopnih podatkov, ki se nahaja v domačem direktoriju uporabnika. Natančneje, konfiguracijski datoteki se nahajata v mapi ".aws", poimenovani pa sta "credentials" in "config".

V datoteki appsettings.json, OpenReportWorkerService projekta, sem dodal sekcijo AWS. Ta je kasneje uporabljena za avtentikacijo zahtev za delo s SQS sporočili.

Delavec deluje tako, da v while zanki, vsake 10 sekund, AWS SQS vrsto vpraša, ali je na voljo kakšno sporočilo, tako, da pošlje vrsti RecieveMessageRequest. Če je v prejetem RecieveMessageResponsu kakšno sporočilo, se za vsako sporočilo ustvari nov tip objekta Note. Vsak se nato z API klicem nato vnese v podatkovno bazo. Pripne se trenutno odprtemu projektu.

Ko je sporočilo vnešeno v podatkovni bazi, se lahko izbriše iz SQS vrste. To dosežemo z DeleteMessageRequest-om.

## 4.8 Postopek oblikovanja in načrtovanja mobilne aplikacije

Mobilno aplikacijo sem se odločil razviti s tehnologijo Xamarin.Forms. Razlogi za to so predvsem dobro poznavanje tehnologije in enostavna integracija z ostalimi tehnologijami iz .NET sklopa. Prav tako se lahko ospredna XAML in ozadna C# koda v veliki meri uporabita v drugih .NET rešitvah.

Postopek sem začel z premislekom o funkcionalnostih aplikacije. Nato sem začel z skico pogledov na list papirja. Ko sem prišel do željene oblike, sem s programom Inkscape narisal prototipe vseh pomembnejših pogledov.

### 4.8.1 Pristop

Pristop razvoja aplikacije, ki sem ga ubral se imenuje MVVM. Pri tem pristopu aplikacijo razdelimo na tri dele.

**Model** je del, kjer definiramo elemente naše poslovne logike, prikaz podatkov, itd.

**View** je uporabniški vmesnik.

**ViewModel** pa se uporablja, da se poveže funkcije uporabniškega vmesnika in poslovne logike.

Rezultat upoštevanja tega pristopa je čista koda. Model vsebuje le abstrakcijo naših podatkov in poslovno logiko. Ti podatki se v ViewModelu pretvorijo v obliko, ki bo prikazana uporabniku. View pa vsebuje izključno UI elemente in povezavo na ViewModel.

// slika mvvm

### 4.8.2 Uvoz knjižnic in zunanjih virov v projekt

Po dizajnu sem se lotil programiranja. Najprej sem ustvaril prazen Xamarin projekt tipa Shell. Najprej sem želel uvoziti funkcionalnosti, ki bi mi v prihodnjih fazah olajšale delo. Prva taka stvar je bila uvoz razširitev Fody. To je orodje, ki med prevajanjem MVVM aplikacije določene odseke C# kode

"vstavi"v preveden assembly. S tem se ohrani ista funkcionalnost z veliko manj boilerplate kode.

// slika fody

Naslednja takšna funkcionalnost je bila uvoz fontov. Za ikone nisem uporabil SVG ali PNG zbirk, ampak sem uporabil popularen font FontAwesome, ki ga nekoliko enostavneje uporabiti in uvoziti.

V projektu OpenReportShell.Android sem v Assets mapo dodal podmapo fonts. V tej mapi se nahaja datoteka s FontAwesome Solid fontom.

Ta font sem uporabil tako, da sem v osnovnem projektu dodal Resource v datoteki App.xaml.

// slika resource

### 4.8.3 Povezava na strežnik in razred Startup.cs

Za povezavo na strežnik sem želel v projektu imeti eno instanco statičnega razreda. Ta razred (OpenReportCloudCommunicationService) vsebuje metode, namenjene komunikaciji z API-jem.

Registriral sem ga po metodi DependencyInjection. To sem dosegel tako, da sem v projekt dodal razred Startup.cs. V tem razredu sem registriral razred ServiceProvider, s katerim bom lahko dostopal do registriranih razredov drugod po aplikaciji.

V ServiceProvider sem dodal dva razreda. Prvi je HttpClient, ki sem ga uporabil za pošiljanje HTTP sporočil na strežnik, drugi pa je OpenReportServerCommunicationService, ki je bil uporabljen za formiranje HTTP sporočil.

// slika ConfigureServices

// slika OpenReportServerCommunicationService

### 4.8.4 DashboardPage

Na strani DashboardPage se uporabniku prikaže seznam odprtih projektov. Ko uporabnik klikne na željeni projekt, ga preusmeri na stran ProjectPage,

kjer ga lahko ureja. Če nanj pridrži, se mu pokažejo dodatne opcije, kot so izbris in ahriviranje.

// slika DashboardPage

### 4.8.5 ProjectPage

Najpomembnejši del aplikacije je urejanje zapiskov. To lahko uporabnik dela,

## 4.9 Evalvacija funkcionalnosti

# Poglavje 5

# Možnosti nadaljnega razvoja

# Poglavje 6

# Zaključek